+

* Computer System

A diagram of a computer system

Description automatically generated

* + Hardware
    - Input ex (RAM , SSD , Disk , Buttons , CPU )
    - Output ex (Monitor, Printer, Speakers, and Projectors)
  + System & App Programs
    - App
      * Programs help me to handle with computers like Compiler.
    - System
      * Programs that users don’t handle with it directly, it helps user to handle with hardware like keyboard when user click on it what the happens.
  + Operating System (software layer)
    - Programs between (System & App Programs) and Hardware
    - Resource allocation
      * It distributes commands (CPU & RAM) to the programs that are running.
  + User View
    - Single User computers (PCs)
      * Ex -> workstations, laptop, Desktop Computer, All-in-one Computer
      * PCs -> Personal Computers
      * All-in-one Computer
        + Computers combine the monitor & CPU.
        + They offer a space-saving design.
    - Multi user computers
      * Ex -> mainframes, computing servers
    - Handheld computers
      * Ex -> Smartphones -> optimized for usability and battery life.
    - Embedded computers
      * Computers in home devices & automobiles
      * Designed primarily to run without user intervention.
      * Ex -> it is in the washing machine.
      * Button performs one function.
  + system View
    - resource allocator
      * Decides between conflicting requests for efficient and fair resource use.
        + (لو عندى اتنين عايزين يستخدموا الشاشه فى نفس الوقت هو اللى بيقرر مين اللى يستخدمها الاول )
    - security
    - Control program
      * Controls execution of programs to prevent errors and improve use of the computer.
  + Kernel -> one program running at all the times on the computer.
    - System programs
      * Programs download with OS (but are not necessarily part of the kernel)
      * Ex -> notepad , google drive.
    - Application programs
      * Programs that are not downloaded with the OS.
      * Ex -> photoshop programs
  + Resulted increase in features
    - Middleware + Kernel
      * a set of software frameworks that provide additional services to application developers.
* Computer System Architecture
  + info
    - how to configure each component
* Computer System Organization
  + info
    - the way programs interact with each other.
  + Modern Computer System
    - CPU.
    - Disk Controller
    - USb controller
    - Graphics adapter
    - memory
  + consists of one or more CPUs and number of Device Controllers connected through a common bus that provides access to shared memory.
  + Each device controller has a local buffer.
  + The CPU & device controllers can execute in parallel.

A computer screen shot of a computer

Description automatically generated

* + Computer Startup

Bootstrap program (Is loaded at power-up or reboot.)

* + - 1. Initializes all aspects of system.
         1. بيصفر الرام اول ما يشتغل
         2. بيصفر البروجرم كونتر
      2. Loads Operating System kernel and start execution.
      3. Typically stored in ROM || EPROM, generally Known as Firmware
         1. ROM -> (مش هتتمسح)
         2. EPROM -> الفوق البنفسجيه هتتمسح بالاشعه))
* I/O Structure
  + Interrupt hardware
    - مثال عليه لما بكتب على الكيبورد وظهور الكتابه يتأخر فكده بتتخزن اللى كتبته فى كونترولير بتاع يو اس بي فهيعمل النترربت للسى بي يو يعنى بيقوله وقف اللى بتعمله وتعالى بس لو شغال فى انترربت تانيه هيقارن على حسب جدول انترربت فيكتورفيها الاولويه وبعد ما يخلص فانكشن ظهور الكلام من الكنترولير هيرجع يكمل اللى كان بيعمله
  + Trap or exception
    - Software-generated interrupt caused either by an error or a user request.

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

NOTES

* An operating system is interrupt driven.
* Passive program -> program still not working.
* Active program (process) -> After right clicking twice.

Overview

* The OS is software that manages computer hardware.
* Because an OS is large & complex, it must be created piece by piece. Each of these piece should be a well-delineated portion of the system with carefully defined inputs, outputs, and functions
* It is intermediate between computer user and computer hardware
* Architecture
  + How to configure each component
  + \*\*\*describes what the computer does
* Organization
  + the way programs interact with each other
  + \*\*\*describes how it does it
* Computer hardware includes the CPU, memory, and I/O devices, as well as storage. A fundamental responsibility of an OS is to allocate these resources to programs
* NOTES

Chapter 1

* 1. What operating system do?
* A computer system can be divided roughly into four components: the hardware, the operating system, Application program & system program, and the user.
* Hardware provides the basic computing resources for the system: CPU, memory, I/O devices, and storage
* Application programs
* Programs that are not downloaded with the OS.
* define the ways in which these are used to solve users` computing programs
* such as word processors, spreadsheets, compiler, and web browsers
* which include all programs not associated with the operation of the system.
* the operating system controls the hardware and coordinates its use among between the various application programs for the various users
* we can also view a computer system as consisting of hardware, software, and data
* OS performs no useful functions by itself. It simply makes programs can do useful work
  + 1. User View
* Some computers have little or no user view
  + Ex -> embedded computers
    1. System View
* An OS is a control program. A control program manages the execution of the user programs to prevent errors & improper use of the computer. Its especially concerned with the operation & control of I/O devices
  + 1. Defining operating systems
* Hardware alone is not particularly easy to use.
* Kernel is a program running at all time on the computer.
* Along with the kernel there are two other types of programs (system programs & application programs).
* System programs
  + which are associated with the OS but are not necessarily part of the kernel.
* Mobile operating systems often include not only a core kernel but also middleware.
* Middleware
  + a set of software frameworks that provide addition services to application developers.
  + ex -> Apple1`s IOS and Google’s Android
* a core kernel along with middleware that supports databases, multimedia, and graphics.
* The OS includes the always running kernel, middleware frameworks that ease application development and provide features, and system programs that aid in managing the system while it`s running.
  1. Computer System Organization

A screenshot of a computer screen

Description automatically generated

* A computer system consists of one or more CPUs and a number of devices controllers connected through a common bus that provides access between components and shared memory.
* Operating systems have a device driver for each device controller.
* Device driver understands the device controller.
* The CPU and the device controllers can execute in parallel.

A computer screen shot of a computer

Description automatically generated

1.2.1) Interrupts

* The device driver loads the appropriate registers in the device controller.
* Hardware may trigger an interrupt at any time by sending a signal to the CPU, usually by way of the system bus.
* Software may trigger on interrupt by executing a special operation called a system call also called a monitor call.
* Interrupt vector contains the addresses of all the service routine.
* A trap or exception is a software-generated interrupt caused either by an error or a user request
* An Operating System is interrupt driven
* There may be many buses within a computer system, but the system bus Is the main communications path between the major components

1.2.1.2) Implementation

* The CPU hardware has a wire called the interrupt-request line

1.2.2) Storage Structure

* The CPU can load instruction sonly from memory, so any programs must first be loaded into memory to run.
* Main memory commonly is implemented in a semiconductor technology called Dynamic Random Access Memory (DRAM)
* The first program to run on computer power-on is a bootstrap program which then loads the operating system and it is storaged in Electrical Erasable Programming Read Only Memory (EEPROM)
* The iPhone uses EEPROM to store serial numbers
* All other storage in a computer is based on collections of bits and hardware information about the device
* Most computers don’t have an instruction to move a bit but do have one to move a byte.
* A computer executes many operations in its native word size rather than a byte at a time.
* Networks move data a bit at a time.
* All forms of memory provide an array of bytes, Each byte has its own address
* Main memory is usually too small to store all needed programs and data permanently.
* The most common secondary-storage devices are hard-disk drivers (HDDs) and non-volatile memory (NVM) devices.
* Most programs (Application & System) are stored in secondary storage until they are loaded into memory.
* The main differences among the various storage systems lie in speed, size, and volatility.

A computer screen shot of a diagram

Description automatically generated

* The most common form NVM device is flash memory, which is popular in mobile devices such as smartphones and tablets
* NVM retains its contents when power is lost. It can be classified into two distinct types
  + mechanical -> HDDs, optical disks, Holographic storage, and magnetic top.
  + Electrical -> flash memory, FRAM, NRAM, and SSD

1.3) Computer-System architecture

* The CPU can load

1.3.1) Single-Processor Systems

* Most computer systems used a single-processor containing one CPU with a single processing core

1.3.2) Multiprocessor Systems

* The benefit of this model is that many processor can run simultaneously

1.3.3) Clustered Systems

* 1. Operating System operations
* Bootstrap program is stored within the computer hardware in firmware, it initialize all aspects of the system, from CPU registers to device controllers to memory content
* The bootstrap program must know how to load the operating system and how to start executing that system.
* Hadoop
  + It is an open-source software framework that is used for distributed processing of large data (big data) in clustered system, low cost hardware components
  + It is designed to scale from a single system to a cluster containing thousands of computing nodes.
  + Detects and manages failures in node.
  + It is designed to run on Linux system
  + Applications can be written using several programming languages
  + It has several Java libraries that support map Reduce
* The bootstrap program must locate the OS kernel and load it, into memory.
* Some services are provided outside of the kernel by system programs that are loaded into memory at boot time to become system daemons
* On Linux, the first system program is system and it starts many other daemons, once this phase is complete, the system waits for some event to occur

1.4.1) Multiprocessing & Multitasking

* The process may have to wait for some tasks, such as an I/O operation, to complete. In a non-multiprogramming system the CPU would sit idle . in a multiprogramming system, the operating system simply switches too, and executes, another process, when that process needs to wait, the CPU switches to another process and soon. Eventually, the first process finishes waiting and gets the CPU back
* Multitasking is a logical extension of multiprogramming
* In multitasking the CPU executes multiple processes by switching among them, but the switches occur frequently

1.4.2) Dual-mode & multimode Operation

* We need two separate modes of operation user mode & kernel mode (supervisor mode , system mode or privileged mode)

A screenshot of a computer

Description automatically generated

* Whenever a trap or interrupt occurs, the hardware switches from user mode to kernel mode, changes the state of the mode bit to zero.

1.5) Resource Management

* The system CPU, memory space, file-storage space, and I/O devices are among the resources that the OS must manage

1.5.1) Process Management

* Jh

1.5.2) Memory Management

* We need two

1.5.3) File System Management

* We need two

1.5.4) Mass-Storage Management

* We need two

1.5.5) Cache Management

* We need two

1.5.6) I/O System Management

* We need two

1.6) Security & Protection

* We need two

1.7) Virtualization

* A user of a virtual machine can switch among the various operating systems in the various processes running concurrently in a single operating system
* Allows operating systems to run a application within other operating systems
* Virtualization software is one member of a class that also includes emulation
* Emulation is used when the CPU type is different from the target CPU type
  + Ex ->when APPLE Switched from the IBM power CPU to the Intel x86 CPU for its desktop and laptop computers, it included an emulation facility called Rosetta, which allowed applications compiled for IBM CPU to run an the Intel CPU
* It the source and target CPUs have similar performance levels, the emulated code may run much more slowly than the native code

A screenshot of a computer

Description automatically generated

* The VMM (Virtual Machine Manager) runs the guest operating systems. Manages their resource use, and protects each guest from the others.
  + - * + NOTES
* Os is -> resource allocator & control program & interrupt driven
* Os has -> kernel & middleware & system application & CPUs & device driver
* (نسخه وندوز بتاعتى غير بتاعت زميلى يبقى الكرنل واحد بس ممكن السيستسم و الابليكيشن مختلفين)
* Memory -> RAM, rewritable memory, random access memory
* KB (1024 Bytes), MB (1024 ^ 2), GB (1024 ^3), TB (1024 ^ 4), PB (1024 ^ 5)

Chapter 2

* 1. Operating-System Services

A screenshot of a computer

Description automatically generated

* Note that these services also make the programming task easier for the programmer.
* User interface
  + Almost all operating systems have a user interface (UI).
  + Ex -> GUI, CLI, and Batch
  + a graphical user interface (GUI) is used
  + Mobile systems such as phones and tablets provide a touch-screen interface
  + command-line interface (CLI), which uses text commands and a method for entering them
* Program execution.
  + The system must be able to load a program into memory and to run that program. The program must be able to end its execution, either normally or abnormally (indicating error).
* I/O operations
  + such as reading from a network interface or writing to a file system.
  + users usually cannot control I/O devices directly.
* File-system manipulation.
  + programs need to read and write files and directories.
  + They also need to create and delete them by name, search for a given file, and list file information.
  + Finally, some operating systems include permissions management to allow or deny access to files or directories based on file ownership.
* Communications
  + communication may occur between processes that are executing on the same computer or between processes that are executing on different computer systems tied together by a network.
  + Communications may be implemented via shared memory or message passing
* Error detection.
  + Errors may occur.
    - in the CPU and memory hardware
      * such as a memory error or a power failure
    - in I/O devices
      * such as a parity error on disk, a connection failure on a network, or lack of paper in the printer
    - in the user program
      * such as an arithmetic overflow or an attempt to access an illegal memory location.
  + it might terminate an error-causing process or return an error code to a process for the process to detect and possibly correct.
* Another set of operating-system functions exists not for helping the user but rather for ensuring the efficient operation of the system itself.
  + Resource allocation
    - When there are multiple processes running at the same time, resources must be allocated to each of them.
    - in determining how best to use the CPU, operating systems have CPU-scheduling routines.
  + Logging
    - Usage statistics may be a valuable tool for system administrators who wish to reconfigure the system to improve computing services.
  + Protection and security
    - Protection should not be possible for one process to interfere with the others or with the operating system itself.
    - Security starts with requiring each user to authenticate himself or herself to the system.
  1. User and Operating-System Interface

2.2.1) Command Interpreters

* On systems with multiple command interpreters to choose from
  + on UNIX and Linux systems, a user may choose among several different shells, including the C shell, Bourne-Again shell, Korn shell, and others.
  + The various shells available on UNIX systems can be implemented in two general ways.
    1. the command interpreter itself contains the code to execute the command
       - For example, a command to delete a file
    2. used by UNIX, among other operating systems
       - implements most commands through system programs.
         * In this case, the command interpreter does not understand the command in any way; it merely uses the command to identify a file to be loaded into memory and executed.
* shell command interpreter being used on macOS.
* The various shells available on UNIX systems

2.2.2) Graphical User Interface

* Depending on the mouse pointer’s location
* Various GUI interfaces are available
* Various GUI interfaces are available

2.2.3) Touch-Screen Interface

* users interact by making gestures on the touch screen.
  + for example, pressing and swiping fingers across the screen
* Both the iPad and the iPhone use the Springboard touch-screen interface
  1. System Calls
* Provide an interface to the services made available by an OS.
* These calls are generally available as functions written in C & C++
* May have to be written using assembly-language instructions.
  + Tasks where hardware must be accessed directly.
    1. Example
* The UNIX cp command: cp in.txt out.txt to copy a file from another
  + A screenshot of a computer

    Description automatically generated
    1. Application Programming Interface (API)
* Application developers design programs according to an API
* The API specifies a set of functions that are available to an application programmer, including the parameters that are passed to each function and the return values the programmer can expected.
* Examples
  + The POSIX API for POSIX-based systems (include virtually all versions of UNIX, LINUX, and macOS)
  + Windows API for Windows systems
  + Java API for programs that run on the Java virtual machine.
* A programmer accesses an API via a library of code provided by the operating system.
* UNIX and Linux for programs written in the C language, the library is called libc
* A screenshot of a computer

  Description automatically generated
* For example -> the Windows function CreateProcess() actually invokes the NTCreateOrocess() system call in the windows kernel
* Why would an application programmer prefer programming according to an API rather than invoking actual system calls?
  + An API can expect her program to compile and rrun on any system that supports the same API
  + Actual system calls can often be ,ore detailed and difficult
  + Many of thde POSIX and Windows APIs are similar to the native system calls provided by the UNIX, Linux, and Windows operating System
* Run Time Environment (RTE)
  + The full suite of software needed to execute applications written in a given programming language
  + Including its compilers or interpreters as well as other software, such as libraries and loaders
  + Provides system-call interface that serves as the link to system calls made available by the operating system
  + The system-call interface intercepts function calls in the API ang invokes the necessary system calls within the operating system
  + a number is associated with each system call, and the system-call interface maintains a table indexed according to these numbers. The system-call interface then invokes the intended system call in the operating-system kernel and returns the status of the system call.
  + most of the details of the operating-system interface are hidden from the programmer by the API and are managed by the RTE.